УДК 004.258 DOI: 10.22184/NanoRus.2019.12.89.99.101

ПРОЕКТИРОВАНИЕ ПОДСИСТЕМЫ ПАМЯТИ ДЛЯ ОБЕСПЕЧЕНИЯ БЫСТРОДЕЙСТВИЯ ПРИ РЕШЕНИИ ЗАДАЧ ТРЕХМЕРНОЙ ГРАФИКИ

MEMORY SUBSYSTEM DESIGN TO ENSURE PERFORMANCE IN SOLVING 3D GRAPHICS PROBLEMS

АРЯШЕВ СЕРГЕЙ ИВАНОВИЧ **ARYASHEV SERGEY I.**

КОРНИЛЕНКО АЛЕКСАНДР ВЛАДИМИРОВИЧ **KORNILENKO ALEKSANDER V.**

ЗУБКОВСКИЙ ПАВЕЛ СЕРГЕЕВИЧ **ZUBKOVSKY PAVEL S.**

ЭСУЛА ОЛЬГА ИГОРЕВНА **ESULA OLGA I.**

ФГУ ФНЦ НИИСИ РАН
117218, Москва, Нахимовский просп., 36, к. 1
kerry@cs.niisi.ras.ru

SRISA RAS
bld. 1, 36 Nakhimovskiy Ave., Moscow, 117218
kerry@cs.niisi.ras.ru

В статье рассмотрены характерные требования контроллера трехмерной графики к подсистеме памяти и дано описание структуры микропроцессора на основе архитектуры «КОМДИВ-64». Приведены результаты измерений пропускной способности подсистемы памяти и анализ влияния на быстродействие при решении задач трехмерной графики. Описаны схемотехнические решения для обеспечения технических требований. Представлен способ автономного тестирования этих решений.

Ключевые слова: трехмерная графика; 3D; подсистема памяти; обеспечение быстродействия; пропускная способность памяти.

The article considers typical requirements of the three-dimensional graphics controller for the memory subsystem and describes the structure of the KOMDIV-64 microprocessor. The results of measurements of memory subsystem bandwidth and analysis of the impact on the performance in solving three-dimensional graphics problems have been presented. Circuit solutions for technical requirements have been described. A way to test these solutions has been presented.

Keywords: three-dimensional graphics; 3D; memory subsystem; performance; memory bandwidth.

Микропроцессор (МП) разработки НИИСИ РАН является одноядерным МП общего назначения с RISC-архитектурой «КОМДИВ-64» и встроенным графическим ядром.

На основе опыта проектирования МП предыдущих версий и теоретического анализа технологических ограничений создана модель подсистемы памяти на языке описания аппаратуры Verilog. В статье описаны реализованные в модели схемотехнические решения, позволяющие достичь требуемого быстродействия трехмерной графики. Также в статье представлен автономный способ верификации контроллера памяти, дающий возможность ускорить отладку проекта.

МП спроектирован для изготовления по нормам 65 нм. Тактовая частота процессорного ядра — 0,8 ГГц. На одном кристалле размещены: одно процессорное ядро RISC-архитектуры, контроллер PCI Express для связи с вычислительной системой в случае работы МП в режиме видеоконтроллера со встроенным трехмерным ускорителем и собственно графическое ядро. Внутренняя шина работает по протоколу AMBA AXI3 [1] (далее шина AXI) с шириной слова 128 бит и длиной пакета одной передачи до 16 слов. Структурная схема МП показана на рис. 1.

Ядро микропроцессора является суперскалярным и включает в себя раздельные кеш-памяти первого уровня для инструкций и данных объемом 16 КБ каждая и общую кеш-память второго уровня объемом 512 КБ. В составе ядра имеется сопроцессор вещественных вычислений, а также сопроцессор комплексных вычислений, ориентированный на задачи цифровой обработки сигналов. Пиковая производительность сопроцессора комплексных вычислений составляет 20 вещественных операций одинарной точности за такт и 10 вещественных операций двойной точности за такт.

Графическое ядро состоит из двух контролеров вывода на экран и ускорителей двухмерной и трехмерной графики. В свою очередь, ядро трехмерного ускорителя состоит из двух кластеров, каждый из которых имеет доступ в память по шине AXI. Для обеспечения работы трехмерного ядра на кристалле расположен отдельный синтезатор частоты, не связанный с частотой шины AXI.

При наличии двух шин AXI (количество шин $a = 2$) шириной 128 бит ($d = 128$), работающих на частоте 0,5 ГГц ($f = 0{,}5$), выходит, что максимальная скорость ($v = a \cdot d \cdot f$) передаваемых или требуемых трехмерным ядром данных составляет 16 Гбайт/сек. Контроллер памяти DDR3 шириной 64 разряда, работающий на частоте 1 ГГц, в пике теоретически способен обеспечить также те же 16 Гбайт/сек. При этом графический кластер как вычислитель, как правило, зачитывает данные, в том числе во внутреннюю кеш-память, а затем после обработки выгружает в основную память.

Таким образом, основная задача при проектировании состоит в том, чтобы трехмерный ускоритель получал при

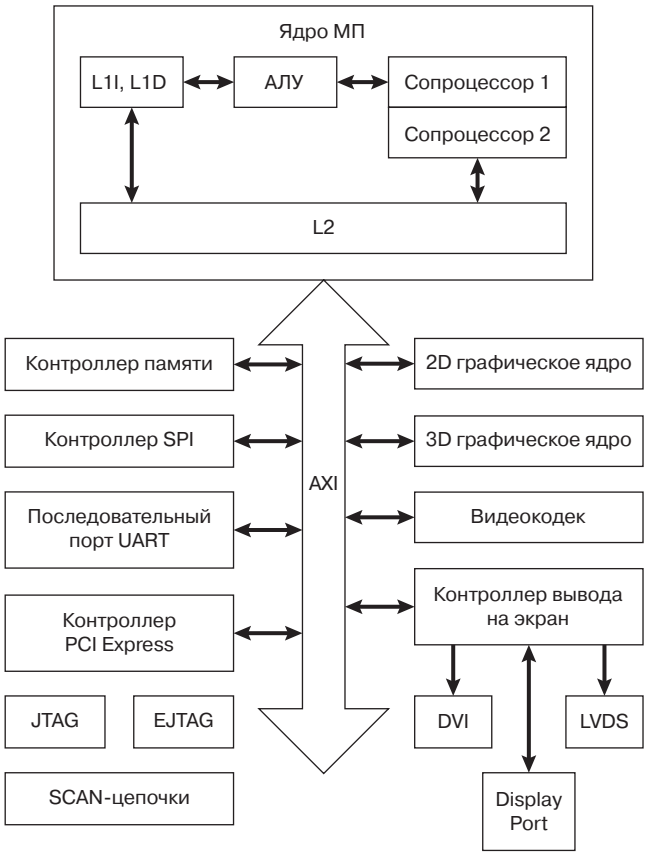

Рис. 1. Структурная схема МП. Сопроцессор 1 — сопроцессор вещественных вычислений; сопроцессор 2 — сопроцессор комплексных вычислений; AXI — внутренняя шина, работающая по протоколу AMBA AXI3

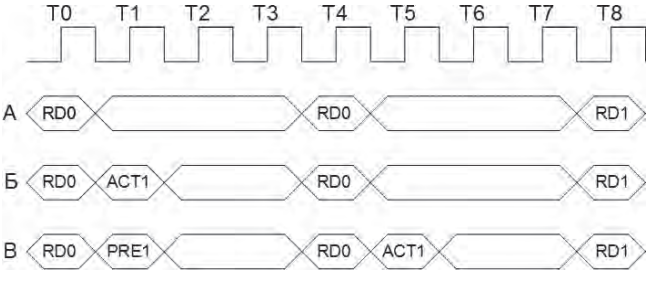

Рис. 2. Варианты вложенности циклов при обращении к памяти

необходимости максимальную пропускную способность шины памяти.

Для достижения поставленной цели — получения максимальной производительности применяются несколько методов:

1) механизм обеспечения гарантированной полосы пропускания (QoS);
2) кеширование данных;
3) минимизация простоя шины данных памяти.

Первый механизм служит для предоставления устройству гарантированной доли от полной полосы пропускания и позволяет устройству не только получить необходимое количество данных за отрезок времени, но и провести времени в ожидании доступа при арбитрации не больше определенного [2]. Данный механизм является стандартным в контроллерах памяти вычислительных систем и применяется для решения не только трехмерных задач, но и любых, где требуется в зависимости от задачи разделить производительность контроллера памяти между разными устройствами.

Кеширование данных производится непосредственно в контроллере трехмерной графики, каждый кластер имеет свой кеш первого уровня, и затем общий контроллер шины AXI также содержит кеш второго уровня. По этой причине другого накопительного механизма на шине AXI не требуется, он не будет давать эффекта.

Простои на шине памяти неизбежны, поскольку по самой архитектуре динамической памяти ей периодически требуется регенерация. При работе в условиях повышенных температур к тому же она должна быть чаще, и пока регенерация текущей строки не закончится, работа с памятью невозможна. Кроме того, после записи в память должно быть выдержано время tWR, за которое также никаких команд на память подавать нельзя.

На рис. 2 показаны возможности повышения быстродействия подсистемы памяти. Времена, которые требуются выдержать между командами PRECHARGE, ACTIVATE и READ, зависят от частоты, на которой работает память, и на рисунке показаны условно.

В случае А мы рассматриваем чтение из памяти без вложенных циклов. Память DDR3 настроена на длину пакета 8, то есть с учетом работы памяти по двум фронтам команда чтения должна выставляться раз в четыре такта. Индексы команд показывают, что операция будет совершаться с тем или иным банком памяти, условно 0 или 1. Если следующее чтение будет из того же банка или из другого, но уже с открытой нужной строкой, то чтения будут идти подряд и пропускная способность памяти будет использоваться максимально. В случае Б показана ситуация, в которой чтение из банка 1 требует активирования строки, из которой будет чтение. Если это сделать после чтения, то будет тратиться время на активацию, во время которой шина данных простаивает. В случае В ситуация усугубляется тем, что в банке открыта другая строка, то есть сперва требуется выполнить закрытие строки в банке, открыть нужную и лишь затем провести чтение. На это нужен еще больший период, за время которого шина данных также простаивает.

В таком случае есть возможность сократить простои, выполняя предварительные действия для следующей операции в момент прохождения предыдущей. Контроллер памяти прослушивает шину, на которую выставляется адрес следующей команды, и принимает решение, возможно ли вставить предварительную команду.

Рассмотрим результаты моделирования различных тестов. Измерялось время активной фазы обращения к данным, то есть от начала доступа к памяти до последней операции с памятью. Как правило, доступ к различным участкам памяти производится в случае копирования, выполняемого в кешируемой области, потому что кроме чтения из области назначения периодически происходит вытеснение устаревающих строк кешей по разным адресам. В качестве теста взяты две процедуры копирования, схожие со стандартными, используемыми ОС Linux, и текст произвольного доступа к памяти. При работе контроллера трехмерной графики операции копирования выполняются постоянно: происходит подготовка текстур, перенос данных из промежуточных буферов в буфер вывода на экран и так далее. Из табл. 1 видно, что копирование может быть существенно ускорено при использовании вложенности циклов, тем самым для контроллера трехмерной графики увеличится время, которое тот может использовать для обмена данных с памятью.

Подобная модификация алгоритма работы контроллера усложняет как его работу, так и поиск возможной ошибки, потому требуется всесторонняя и тщательная проверка его работоспособности не только в ПЛИС-прототипе, но и средствами тестирования на этапе отладки модели.

Проверка корректности поведения контроллера памяти, то есть соответствие поведения модели спецификации, началась с создания группой инженеров, разработчиков проекта и верификаторов, плана тестирования. Формальные утверждения (*assertion*) [3] и функциональное покрытие (*functional coverage*) были написаны на основе плана тестирования, который можно разделить на четыре части:

1. Интерфейсная часть: контроль корректной реализации интерфейсов контроллера памяти, работающий по протоколу AMBA AXI3.
2. Функциональное ядро: описание механизма работы контроллера памяти.
3. Специальные случаи: описание corner case и критических ситуаций.
4. Покрытие кода: code coverage помогает увидеть неиспользуемые участки кода.

План тестирования связал описание контроллера памяти и RTL-модель с помощью assertions и functional coverage. Прогресс тестирования хорошо виден по достижению новых точек functional coverage. Многократное прохождение по одним и тем же точкам functional coverage и отсутствие попадания по другим точкам помогает скорректировать тестовые воздействия. Полное выполнение тестового плана служит объективным критерием окончания процесса тестирования.

Кроме системной верификации применялась автономная. Так как внесенные в проект изменения располагались только в одном устройстве, автономная верификация была приоритетной. Она позволила упростить и ускорить процесс отладки контроллера памяти. Во-первых, не нужно программировать все устройства, обращающиеся к памяти, готовить дескрипторы и данные. Во-вторых, отсутствуют модели всех задействованных в проекте устройств. В-третьих, нет необходимости дожидаться окончания работы устройств, обращающихся в память.

Контроллер памяти имеет семь портов с независимыми каналами чтения и записи, работающим по AMBA AXI3. К каждому порту подсоединен UVM-агент, чье поведение описано с использованием библиотеки последовательностей настраиваемых запросов. В качестве языка реализации выбран Specnam *e*.

При написании UVM-окружения особое внимание уделялось описанию задержек, так как для мастера шины нужна регулировка плотности запросов, а для ведомого устройства необходима регулировка задержки в ответе. В состав логического пакета (в классе-наследнике uvm_sequence_item) включены поля, отвечающие за задержки. В результате они могут контролироваться при генерации запросов. При этом в драйвере шины учитывается возможность ожидания выставления на шину второго пакета, пока не будет полностью обработан первый (re-order не поддерживается устройствами на шине).

Автономное тестирование не учитывает когерентность данных, что позволяет ускорить процесс верификации, разделив области памяти между портами. В тестовом окружении эта особенность учитывается за счет использования двух параметров, формирующих адрес: базовый адрес и смещение. Базовый адрес ссылается на область памяти, определенную для транзакций, смещение — это относительный адрес внутри этой области. Их сумма является адресом транзакции. Подобное формирование адреса позволяет гибко контролировать обращения к памяти.

Таблица 1

Тест	Результат без вложенности, мкс	Результат со вложенностью, мкс	Разница, %
Glibc copy 32 bit	5,9	5,6	5
Glibc copy 64 bit	5,1	4,4	14
Random	22,9	22,1	3

Последовательности настраиваемых запросов определяют входные потоки запросов для тестируемого модуля. Сами последовательности реализованы в виде шаблонов в библиотеке, к которой обращается тест при формировании тестового сценария.

Тест считается успешно прошедшим, если данные по чтению совпали с ранее записанными (или теми, что инициализировали в память) и при этом не было ошибок в ситуациях, описанных assertions. Механизм сравнения данных сделан универсальным для трех значений параметра протокола AXI3 size = 4, 3, 2, так как именно они используются при имитации входных запросов в контроллер памяти.

Более полное покрытие RTL-модели тестами обеспечено использованием виртуальных последовательностей, позволяющих контролировать выбор типа последовательностей запросов и параметры запросов, а также другие настройки тестового сценария.

Моделирование проводилось с помощью симулятора Cadence Incisive Enterprise Simulator. Запуск тестов осуществлялся программой Vmanager, входящей в комплект поставки. Vmanager позволяет контролировать ошибочные тесты и регрессионное тестирование, а также собрать и объединить functional coverage для различных запусков. На основе анализа собранной информации менялись ограничения для запросов и сценарии тестов, что позволило ускорить процесс отладки RTL-модели контроллера памяти.

Авторы считают, что в представленной в статье новой является RTL-модель контроллера памяти, написанная применительно к решению определенной задачи, а именно повышению эффективности работы с контроллером трехмерной графики, использующей шину по протоколу AMBA AXI3.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме №0065-2019-0004.

ЛИТЕРАТУРА

1. AMBA® AXI™ and ACE™ Protocol Specification // ARM, 2011. P. 28–33.
2. Корниленко А. В., Эсула О. И. Оптимизация подсистемы памяти вычислительной системы с помощью предоставления гарантированной полосы пропускания канала памяти // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС), 2016. — № 3. — С. 136–140.
3. Clifford E. *Cummings. SystemVerilog Assertions — Bindfiles & Best Known Practices for Simple SVA Usage* // SNUG, 2016.